Parallel verification and enumeration of tournaments

László Szűcs and Gábor Pécsy

Round-robin tournaments are popular in the world of sport and they are very much discussed in informatics as well. A round-robin **tournament** is an $n \times n$ real matrix $T_n = [t_{ij}]$ $(n \geq 2)$. The elements of the main diagonal t_{ii} equal to zero and the pairs of symmetric elements $t_{ij} : t_{ji}$ give the result of the match between P_i (the i-th player) and P_j. $t_{ij} = t_{ji}$ means a draw, while $t_{ij} > t_{ji}$ means the win of P_i against P_j. The sum of the elements of the i-th row (s_i) is called the **score** of the i-th player and the vector $(s_1, \ldots, s_n)$ is called the **score vector** of the tournament. A non-decreasingly ordered vector of the scores is denoted $q =< q_1, \ldots, q_n >$ and is called the **score sequence** of the tournament. There are special subsets of tournaments defined by different constraint on the possible results of the matches. We call a set of tournaments **k-complete** if in all matrices all elements are non-negative integers and the sum of the symmetric elements $(t_{ij} + t_{ji},$ where $i \neq j)$ is always k.

The most usually discussed problems regarding tournaments include:

- **Verification of a score sequence/score vector** means the decision if there exists a tournament for a given score sequence/score vector.
- **Enumeration of score sequences** means the counting of the possible different score sequences for a given n number of players.

Several algorithms has been published to solve the above mentioned problems for different subsets of tournaments. One of the most extensively discussed subset is the 1-complete. In this article we present optimal sequential algorithms for all three problems on 1-complete tournaments. We also present their extensions to different parallel architectures including CREW PRAM, linear array, mesh and hypercube. We call a parallel algorithm **work optimal** compared to a given sequential algorithm if $S_n/(P_n * p) = O(1)$ where S_n is the runtime of the sequential algorithm, P_n is the runtime of the parallel algorithm and p is the number of processors. We will show that most of the parallel algorithms presented here are work optimal extensions of the sequential ones.

The table below summarises our results:

Problem	**Sequential**	**Linear array**	**Mesh**	**Hypercube**	**PRAM**
Score sequence	$\Theta(n)$	$\forall p \in \mathbb{N}$ $\Theta(n)$	$p = n$ $O(\sqrt{n})$	$p = \frac{n}{\log n}$ $\Theta(\log n)$ work opt.	$p = \frac{n}{\log n}$ $\Theta(\log n)$ work opt.
Score vector	$\Theta(n)$	$\forall p \in \mathbb{N}$ $\Theta(n)$	$p = n$ $O(\sqrt{n})$	$p = n$ $O(\log^2 n)$	$p = n$ $O(\log^2 n)$
				$p = n^2$ $O(\log n)$	$p = n^2$ $O(\log n)$
Enumeration of score sequences	Recursive formula with dynamic programming: $\Theta(n^4)$	$p = n$ $\Theta(n^3)$ work opt.	$p = n$ $\Theta(n^3)$ work opt.	$p = n$ $\Theta(n^3)$ work opt.	$p = n$ $\Theta(n^3)$ work opt.
		$p = n^2$ $\Theta(n^2)$	$p = n^2$ $\Theta(n^2)$	$p = n^2$ $\Theta(n^2)$	$p = n^2$ $\Theta(n^2)$
				$p = \frac{n^4}{\log n}$ $O(n * \log n)$	$p = \frac{n^4}{\log n}$ $O(n * \log n)$